How to Ensure a Faithful Polynomial Evaluation with the Compensated Horner Algorithm

Philippe Langlois, Nicolas Louvet
Université de Perpignan, DALI Research Team
$\{$*langlois, nicolas.louvet*$\}$*@univ-perp.fr*

Abstract

The compensated Horner algorithm improves the accuracy of polynomial evaluation in IEEE-754 floating point arithmetic: the computed result is as accurate as if it was computed with the classic Horner algorithm in twice the working precision. Since the condition number still governs the accuracy of this computation, it may return an arbitrary number of inexact digits. We address here how to compute a faithfully rounded result, that is one of the two floating point neighbors of the exact evaluation. We propose an a priori *sufficient condition on the condition number to ensure that the compensated evaluation is faithfully rounded. We also propose a validated and dynamic method to test at the running time if the compensated result is actually faithfully rounded. Numerical experiments illustrate the behavior of these two conditions and that the associated running time over-cost is really interesting.*

1 Introduction

1.1 Motivation

Horner's rule is the classic algorithm when evaluating a polynomial $p(x)$. When performed in floating point arithmetic this algorithm may suffer from (catastrophic) cancellations and so yields a computed value with less exact digits than what the computing precision provides. The relative accuracy of such computed value $\widehat{p}(x)$ verifies the well known following inequality,

$$\frac{|p(x) - \widehat{p}(x)|}{|p(x)|} \leq \mathrm{cond}(p, x) \times \mathcal{O}(\mathbf{u}). \qquad (1)$$

In the right-hand side of this accuracy bound, $\mathbf{u}$ is the computing precision and the condition number $\mathrm{cond}(p, x)$ is a scalar larger than 1 that only depends on the entry x and on p coefficients —its expression will be given further. This condition number governs the accuracy of the computed result. It describes the largest magnification factor of the rounding errors both in the data (here, the entry x and the coefficients of p) and in the algorithm (up to now, the Horner Algorithm).

In this paper, we will only consider entries and polynomial coefficients that are floating point values. Such cases appear for example when evaluating elementary functions [11] and in geometric computations [8]. Even in these cases, the product $\mathrm{cond}(p, x) \times \mathcal{O}(\mathbf{u})$ may be arbitrarily larger than 1, *i.e.*, when evaluating the polynomial p at the x entry is ill-conditioned, as for example in the neighborhood of a multiple root.

When the computing precision $\mathbf{u}$ is not sufficient (compared to $\mathrm{cond}(p, x)$) to guarantee a desired accuracy in $\widehat{p}(x)$, several solutions implementing a computation with more bits exist. Priest-like "double-double" algorithms are well-known and well-used solutions to simulate twice the IEEE-754 double precision [13, 9]. The compensated Horner algorithm introduced in [3] is a fast alternative to the Horner algorithm implemented with "double-double" arithmetic. By fast we mean that this compensated algorithm runs at least twice as fast as the "double-double" counterpart with the same output accuracy. In both cases the accuracy of computed $\widehat{p}(x)$ is improved and now verifies

$$\frac{|p(x) - \widehat{p}(x)|}{|p(x)|} \leq \mathbf{u} + \mathrm{cond}(p, x) \times \mathcal{O}(\mathbf{u}^2). \qquad (2)$$

Comparing to Relation (1), this relation means that the computed value is now as accurate as the result of the Horner algorithm performed in twice the working precision with a final rounding back to this working precision —the same behavior is mentioned in [12] for compensated summation and dot product. As for Relation (1) the accuracy of the compensated result still depends on the condition number and may be arbitrarily bad for ill-conditioned polynomial evaluations. Nevertheless, this bound tells us that the compensated Horner algorithm may yield a full precision accuracy for not too ill-conditioned polynomials, that is for p and x such that the second term $\mathrm{cond}(p, x) \times \mathcal{O}(\mathbf{u}^2)$ is small compared to the working precision $\mathbf{u}$.

This remark motivates this paper where we consider how to compute a *faithfully rounded polynomial evaluation* with the compensated Horner algorithm. By faithful rounding we mean that the computed result $\widehat{p}(x)$ is one of the two floating point neighbors of the exact result $p(x)$. Faithful rounding is known to be an interesting property since for example it guarantees the correct sign determination of arithmetic expressions, *e.g.*, for geometric predicates.

We first provide an *a priori* sufficient criterion we summarize as follows. The compensated Horner algorithm provides a faithful rounding of the exact polynomial evaluation as long as its condition number is less than the upper bound we identify in Theorem 7; this bound only depends on the degree of the polynomial and on the working precision $\mathbf{u}$. With Theorem 9, we also propose a dynamic test to answer to the following question at the running time: "is the computed compensated result a faithful rounding of the exact evaluation?". If this test is satisfied then the answer is "yes". Otherwise, the computed result may or may not be faithfully rounded. Moreover, this test is validated since it takes into account the finite precision of its computation performed in IEEE-754 floating point arithmetic. Numerical experiments show that the dynamic bound is sharper than the *a priori* condition and that the corresponding overcost is reasonable.

The paper is organized as follows. In the sequel of this section, we briefly recall the standard model of floating point arithmetic and we introduce our notations for error analysis. In Section 2, we first recall some well known facts about the Horner algorithm and we review the error-free transformations of the elementary floating point operations $+$, $-$ and $\times$. Next we derive an error-free transformation for the Horner algorithm that allows us to compute the compensated evaluation. We prove a theoretical error bound for the accuracy obtained thanks to the compensated Horner algorithm in Section 3, and we describe the *a priori* criterion we propose to ensure a faithful rounding. Section 4 is devoted to the description of the dynamic counterpart of this *a priori* criterion. Finally in Section 5 we present numerical experiments to exhibit both the numerical behavior and the practical efficiency of our algorithms.

1.2 Notations

Throughout the paper, we assume a floating point arithmetic adhering to the IEEE-754 floating point standard [5]. We constraint all the computations to be performed in one working precision, with the "round to the nearest" rounding mode. We also assume that no overflow nor underflow occurs during the computations. Next notations are standard (see [4, chap. 2] for example). $\mathbb{F}$ is the set of all normalized floating point numbers and $\mathbf{u}$ denotes the unit roundoff, that is half the spacing between 1 and the next representable floating point value. For IEEE-754 double precision with rounding to the nearest, we have $\mathbf{u} = 2^{-53} \approx 1.11 \cdot 10^{-16}$. We define the floating point predecessor and successor of a real number r as $\mathrm{pred}(r) = \max\{f \in \mathbb{F}/f < r\}$ and $\mathrm{succ}(r) = \min\{f \in \mathbb{F}/r < f\}$ respectively. A floating point number f is defined to be a faithful rounding of a real number r if $\mathrm{pred}(f) < r < \mathrm{succ}(f)$.

The symbols $\oplus$, $\ominus$, $\otimes$ and $\oslash$ represent respectively the floating point addition, subtraction, multiplication and division. For more complicated arithmetic expressions, $\mathrm{fl}(\cdot)$ denotes the result of a floating point computation where every operation inside the parenthesis is performed in the working precision. So we have for example, $a \oplus b = \mathrm{fl}(a + b)$.

When no underflow nor overflow occurs, the following standard model describes the accuracy of every considered floating point computation. For two floating point numbers a and b and for $\circ$ in $\{+, -, \times, /\}$, the floating point evaluation $\mathrm{fl}(a \circ b)$ of $a \circ b$ is such that

$$\mathrm{fl}(a \circ b) = (a \circ b)(1 + \varepsilon_1) = (a \circ b)/(1 + \varepsilon_2), \quad \text{with} \quad |\varepsilon_1|, |\varepsilon_2| \le \mathbf{u}. \tag{3}$$

To keep track of the $(1+\varepsilon)$ factors in next error analysis, we use the classic $(1 + \theta_k)$ and γ_k notations [4, chap. 3]. For any positive integer k, θ_k denotes a quantity bounded according to $|\theta_k| \le \gamma_k := k\mathbf{u}/(1 - k\mathbf{u})$. When using these notations, we always implicitly assume $k\mathbf{u} < 1$. In further error analysis, we essentially use the following relations,

$$(1 + \theta_k)(1 + \theta_j) \le (1 + \theta_{k+j}), \quad k\mathbf{u} \le \gamma_k, \quad \gamma_k \le \gamma_{k+1}.$$

Next bounds are computable floating point values that will be useful to derive dynamic validation in Section 4. We denotes $\mathrm{fl}(\gamma_k) = (k\mathbf{u}) \oslash (1 \ominus k\mathbf{u})$ by $\widehat{\gamma}_k$. We know that $\mathrm{fl}(k\mathbf{u}) = k\mathbf{u} \in \mathbb{F}$, and $k\mathbf{u} < 1$ implies $\mathrm{fl}(1 - k\mathbf{u}) = 1 - k\mathbf{u} \in \mathbb{F}$. So $\widehat{\gamma}_k$ only suffers from a rounding error in the division and

$$\gamma_k \le (1 + \mathbf{u})\widehat{\gamma}_k. \tag{4}$$

The next bound comes from the direct application of Relation (3). For $x \in \mathbb{F}$ and $n \in \mathbf{N}$,

$$(1 + \mathbf{u})^n |x| \le \mathrm{fl}\left(\frac{|x|}{1 - (n+1)\mathbf{u}}\right). \tag{5}$$

2 From Horner to compensated Horner algorithm

The compensated Horner algorithm improves the classic Horner iteration computing a correcting term to compensate the rounding errors the classic Horner iteration generates in floating point arithmetic. Main results about compensated Horner algorithm are summarized in this section; see [3] for a complete description.

2.1 Polynomial evaluation and Horner algorithm

The classic condition number of the evaluation of $p(x) = \sum_{i=0}^{n} a_i x^i$ at a given entry x is [2]

$$\mathrm{cond}(p,x) = \frac{\sum_{i=0}^{n} |a_i||x|^i}{|\sum_{i=0}^{n} a_i x^i|} := \frac{\widetilde{p}(x)}{|p(x)|}. \qquad (6)$$

For any floating point value x we denote by $\mathsf{Horner}\,(p,x)$ the result of the floating point evaluation of the polynomial p at x using next classic Horner algorithm.

Algorithm 1 Horner algorithm

```
function $r_0 = \mathsf{Horner}\,(p,x)$
$r_n = a_n$
for $i = n-1 : -1 : 0$
    $r_i = r_{i+1} \otimes x \oplus a_i$
end
```

The accuracy of Algorithm 1 verifies introductory inequality (1) with $\mathcal{O}(\mathbf{u}) = \gamma_{2n}$ and previous condition number (6). Clearly, the condition number $\mathrm{cond}(p,x)$ can be arbitrarily large. In particular, when $\mathrm{cond}(p,x) > 1/\gamma_{2n}$, we cannot guarantee that the computed result $\mathsf{Horner}\,(p,x)$ contains any correct digit.

We further prove that the error generated by the Horner algorithm is exactly the sum of two polynomials with floating point coefficients. The next lemma gives bounds of the generated error when evaluating this sum of polynomials applying the Horner algorithm.

Lemma 1. *Let p and q be two polynomials with floating point coefficients, such that $p(x) = \sum_{i=0}^{n} a_i x^i$ and $q(x) = \sum_{i=0}^{n} b_i x^i$. We consider the floating point evaluation of $(p+q)(x)$ computed with $\mathsf{Horner}\,(p \oplus q, x)$. Then, in case no underflow occurs, the computed result satisfies the following forward error bound,*

$$|(p+q)(x) - \mathsf{Horner}\,(p \oplus q, x)\,| \leq \gamma_{2n+1}(\widetilde{p+q})(x). \quad (7)$$

Moreover,

$$(|p+q|)(|x|) \leq (1+\mathbf{u})^{2n+1}\mathsf{Horner}\,(|p \oplus q|, |x|)\,. \quad (8)$$

Proof. The proof of the error bound (7) is easily adapted from the one of the Horner algorithm (see [4, p.95] for example). To prove (8) we consider Algorithm 1, where

$$r_n = |a_n \oplus b_n| \quad \text{and} \quad r_i = r_{i+1} \otimes x \oplus |a_i \oplus b_i|,$$

for $i = n-1, \ldots, 0$. Then, using the standard model (3) it is easily proved by induction that, for $i = 0, \ldots, n$,

$$\sum_{j=0}^{i} |a_{n-i+j} + b_{n-i+j}||x^j| \leq (1+\mathbf{u})^{2i+1}|r_{n-i}|, \quad (9)$$

which in turn proves (8) for $i = n$. □

2.2 EFT for the elementary operations

Now we review well known results concerning error free transformation (EFT) of the elementary floating point operations $+$, $-$ and $\times$.

Let $\circ$ be an operator in $\{+,-,\times\}$, a and b be two floating point numbers, and $\widehat{x} = \mathrm{fl}(a \circ b)$. Then their exist a floating point value y such that $a \circ b = \widehat{x} + y$. The difference y between the exact result and the computed result is the rounding error generated by the computation of $\widehat{x}$. Let us emphasize that this relation between four floating point values relies on real operators and exact equality, *i.e.*, not on approximate floating point counterparts. Ogita *et al.* [12] name such a transformation an error free transformation (EFT).

For the EFT of the addition we use the well known TwoSum algorithm by Knuth [6, p.236] that requires 6 flop (floating point operations). $\mathsf{TwoProd}$ by Veltkamp and Dekker [1] performs the EFT of the product and requires 17 flop. The next theorem exhibits the previously announced properties of TwoSum and $\mathsf{TwoProd}$.

Theorem 2 ([12]). *Let a, b in $\mathbb{F}$ and $x, y \in \mathbb{F}$ such that $[x,y] = \mathsf{TwoSum}(a,b)$. Then, ever in the presence of underflow,*

$$a+b = x+y, \quad x = a \oplus b, \quad |y| \leq \mathbf{u}|x|, \quad |y| \leq \mathbf{u}|a+b|.$$

Let $a, b \in \mathbb{F}$ and $x, y \in \mathbb{F}$ such that $[x,y] = \mathsf{TwoProd}(a,b)$. Then, if no underflow occurs,

$$a \times b = x+y, \quad x = a \otimes b, \quad |y| \leq \mathbf{u}|x|, \quad |y| \leq \mathbf{u}|a \times b|.$$

We notice that algorithms TwoSum and $\mathsf{TwoProd}$ only require well optimizable floating point operations. They do not use branches, nor access to the mantissa that can be time-consuming. We just mention that a significant improvement of $\mathsf{TwoProd}$ is defined when a Fused-Multiply-and-Add operator is available [10].

2.3 An EFT for the Horner algorithm

As previously mentioned, next EFT for the polynomial evaluation with the Horner algorithm exhibits the exact rounding error generated by the Horner algorithm together with an algorithm to compute it.

Theorem 3 ([3]). *Let $p(x) = \sum_{i=0}^{n} a_i x^i$ be a polynomial of degree n with floating point coefficients, and let x be a floating point value. Then Algorithm 2 computes both $\mathsf{Horner}\,(p,x)$ and two polynomials p_π and p_σ of degree $n-1$ with floating point coefficients, such that $[\mathsf{Horner}\,(p,x)\,, p_\pi, p_\sigma] = \mathsf{EFTHorner}\,(p,x)$. If no underflow occurs,*

$$p(x) = \mathsf{Horner}\,(p,x) + (p_\pi + p_\sigma)(x). \qquad (10)$$

Algorithm 2 EFT for the Horner algorithm

```
function $[s_0, p_\pi, p_\sigma] = \mathsf{EFTHorner}(p, x)$
$s_n = a_n$
for $i = n-1 : -1 : 0$
    $[p_i, \pi_i] = \mathsf{TwoProd}\,(s_{i+1}, x)$
    $[s_i, \sigma_i] = \mathsf{TwoSum}\,(p_i, a_i)$
    Let $\pi_i$ be the coefficient of degree $i$ in $p_\pi$
    Let $\sigma_i$ be the coefficient of degree $i$ in $p_\sigma$
end
```

Moreover,

$$\widetilde{(p_\pi + p_\sigma)}(x) \leq \gamma_{2n}\,\widetilde{p}(x). \tag{11}$$

Relation (10) means that algorithm EFTHorner is an EFT for polynomial evaluation with the Horner algorithm.

Proof of Theorem 3. Since TwoProd and TwoSum are EFT from Theorem 2 it follows that $s_{i+1}x = p_i + \pi_i$ and $p_i + a_i = s_i + \sigma_i$. Thus we have $s_i = s_{i+1}x + a_i - \pi_i - \sigma_i$, for $i = 0, \ldots, n-1$. Since $s_n = a_n$, at the end of the loop we have

$$s_0 = \sum_{i=0}^{n} a_i x^i - \sum_{i=0}^{n-1} \pi_i x^i - \sum_{i=0}^{n-1} \sigma_i x^i,$$

which proves (10).

Now we prove relation (11) According to the error analysis of the Horner algorithm (see [4, p.95]), we can write

$$\mathsf{Horner}\,(p, x) = (1 + \theta_{2n}) a_n x^n + \sum_{i=0}^{n-1} (1 + \theta_{2i+1}) a_i x^i,$$

where every θ_k satisfies $|\theta_k| \leq \gamma_k$. Then using (10) we have

$$(p_\pi + p_\sigma)(x) = -\theta_{2n} a_n x^n - \sum_{i=0}^{n-1} \theta_{2i+1} a_i x^i.$$

It yields

$$\widetilde{(p_\pi + p_\sigma)}(x) \leq \gamma_{2n}|a_n||x|^n + \sum_{i=0}^{n-1} \gamma_{2i+1}|a_i||x|^i \leq \gamma_{2n}\,\widetilde{p}(x),$$

hence $\widetilde{(p_\pi + p_\sigma)}(x) \leq \gamma_{2n}\,\widetilde{p}(x)$. □

2.4 Compensated Horner algorithm

From Theorem 3 the forward error in the floating point evaluation of $p(x)$ with the Horner algorithm is

$$c = p(x) - \mathsf{Horner}\,(p, x) = (p_\pi + p_\sigma)(x),$$

where the two polynomials p_π and p_σ are exactly identified by EFTHorner (Algorithm 2) —this latter also computes Horner (p, x). Therefore, the key of the compensated algorithm is to compute, in the working precision, first an approximate $\widehat{c}$ of the final error c and then a corrected result

$$\overline{r} = \mathsf{Horner}\,(p, x) \oplus \widehat{c}.$$

These two computations lead to next compensated Horner algorithm CompHorner (Algorithm 3).

Algorithm 3 Compensated Horner algorithm

```
function $\overline{r} = \mathsf{CompHorner}\,(p, x)$
$[\widehat{r}, p_\pi, p_\sigma] = \mathsf{EFTHorner}\,(p, x)$
$\widehat{c} = \mathsf{Horner}\,(p_\pi \oplus p_\sigma, x)$
$\overline{r} = \widehat{r} \oplus \widehat{c}$
```

We say that $\widehat{c}$ is a correcting term for Horner (p, x). The corrected result $\overline{r}$ is expected to be more accurate than the first result Horner (p, x) as proved in next section.

3 *A priori* condition for faithful rounding

We start proving the accuracy of the compensated Horner algorithm as the first step towards an *a priori* sufficient condition for a faithful rounding.

3.1 Accuracy of CompHorner

Next result proves that the result of a polynomial evaluation computed with the compensated Horner algorithm (Algorithm 3) is as accurate as if computed by the classic Horner algorithm using twice the working precision and then rounded to the working precision.

Theorem 4 ([3]). *Consider a polynomial p of degree n with floating point coefficients, and x a floating point value. If no underflow occurs,*

$$|\mathsf{CompHorner}\,(p, x) - p(x)| \leq \mathbf{u}|p(x)| + \gamma_{2n}^2\,\widetilde{p}(x). \tag{12}$$

Proof. The absolute forward error generated by Algorithm 3 is

$$\begin{aligned} |\,\overline{r} - p(x)| &= |(\,\widehat{r} \oplus \widehat{c}) - p(x)| \\ &= |(1 + \varepsilon)(\,\widehat{r} + \widehat{c}) - p(x)|, \end{aligned}$$

with $|\varepsilon| \leq \mathbf{u}$. Let $c = (p_\pi + p_\sigma)(x)$. From Theorem 3 we have $\widehat{r} = \mathsf{Horner}\,(p, x) = p(x) - c$, thus

$$\begin{aligned} |\,\overline{r} - p(x)| &= |(1 + \varepsilon)\,(p(x) - c + \widehat{c}) - p(x)| \\ &\leq \mathbf{u}|p(x)| + (1 + \mathbf{u})|\,\widehat{c} - c|. \end{aligned}$$

Since $\widehat{c} = \mathsf{Horner}\,(p_\pi \oplus p_\sigma, x)$ with p_π and p_σ two polynomials of degree $n-1$, Lemma 1 yields $|\,\widehat{c} - c| \leq \gamma_{2n-1}\widetilde{(p_\pi + p_\sigma)}(x)$. Then using (11) we have $|\,\widehat{c} - c| \leq \gamma_{2n-1}\gamma_{2n}\,\widetilde{p}(x)$. Since $(1+\mathbf{u})\gamma_{2n-1} \leq \gamma_{2n}$, we finally write the expected error bound (12). □

Remark 1. For later use, we notice that $|\widehat{c} - c| \leq \gamma_{2n-1}\gamma_{2n}\,\widetilde{p}(x)$ implies

$$|\widehat{c} - c| \leq \gamma_{2n}^2\,\widetilde{p}(x). \tag{13}$$

It is interesting to interpret the previous theorem in terms of the condition number of the polynomial evaluation of p at x. Combining the error bound (12) with the condition number (6) of polynomial evaluation gives the precise writing of our introductory inequality (2),

$$\frac{|\mathsf{CompHorner}\,(p,x) - p(x)|}{|p(x)|} \leq \mathbf{u} + \gamma_{2n}^2\,\mathrm{cond}(p,x). \tag{14}$$

In other words, the bound for the relative error of the computed result is essentially γ_{2n}^2 times the condition number of the polynomial evaluation, plus the inevitable summand $\mathbf{u}$ for rounding the result to the working precision. In particular, if $\mathrm{cond}(p,x) < \mathbf{u}/\gamma_{2n}^2$, then the relative accuracy of the result is bounded by a constant of the order $\mathbf{u}$. This means that the compensated Horner algorithm computes an evaluation accurate to the last few bits as long as the condition number is smaller than $\mathbf{u}/\gamma_{2n}^2 \approx 1/4n^2\mathbf{u}$. Besides that, relation (14) tells us that the computed result is as accurate as if computed by the classic Horner algorithm with twice the working precision, and then rounded to the working precision.

3.2 *A priori* condition for faithful rounding

Now we propose a sufficient condition on $\mathrm{cond}(p,x)$ to ensure that the corrected result $\overline{r}$ computed with the compensated Horner algorithm is a faithful rounding of the exact result $p(x)$.We use the following lemma from [14].

Lemma 5 ([14]). *Let r, δ be two real numbers and $\overline{r} = \mathrm{fl}(r)$. We assume here that $\overline{r}$ is a normalized floating point number. If $|\delta| < \frac{\mathbf{u}}{2}|\overline{r}|$ then $\overline{r}$ is a faithful rounding of $r + \delta$.*

From Lemma 5, we derive a useful criterion to ensure that the compensated result provided by CompHorner is faithfully rounded to the working precision.

Lemma 6. *Let p be a polynomial of degree n with floating point coefficients, and x be a floating point value. We consider the approximate $\overline{r}$ of $p(x)$ computed with* CompHorner (p,x)*, and we assume that no underflow occurs during the computation. Let c denotes $c = (p_\pi + p_\sigma)(x)$. If $|\widehat{c} - c| < \frac{\mathbf{u}}{2}|\overline{r}|$, then $\overline{r}$ is a faithful rounding of $p(x)$.*

Proof. We assume that $|\widehat{c} - c| < \frac{\mathbf{u}}{2}|\overline{r}|$. From the notations of Algorithm 3, we recall that $\mathrm{fl}(\widehat{r} + \widehat{c}) = \overline{r}$. Then from Lemma 5 it follows that $\overline{r}$ is a faithful rounding of $\widehat{r} + \widehat{c} + c - \widehat{c} = \widehat{r} + c$. Since $[\widehat{r}, p_\pi, p_\sigma] =$ EFTHorner (p,x), Theorem 3 yields $p(x) = \widehat{r} + c$. Therefore $\overline{r}$ is a faithful rounding of $p(x)$. □

The criterion proposed in Lemma 6 concerns the accuracy of the correcting term $\widehat{c}$. Nevertheless Relation (13) pointed after the proof of Theorem 4 says that the absolute error $|\widehat{c} - c|$ is bounded by $\gamma_{2n}^2\,\widetilde{p}(x)$. This provides us a more useful criterion, since it relies on the condition number $\mathrm{cond}(p,x)$, to ensure that CompHorner computes a faithfully rounded result.

Theorem 7. *Let p be a polynomial of degree n with floating point coefficients, and x a floating point value. If*

$$\mathrm{cond}(p,x) < \frac{1-\mathbf{u}}{2+\mathbf{u}}\mathbf{u}{\gamma_{2n}}^{-2}, \tag{15}$$

then CompHorner (p,x) *computes a faithful rounding of the exact $p(x)$.*

Proof. We assume that (15) is satisfied and we use the same notations as in Lemma 6. First we notice that $\overline{r}$ and $p(x)$ are of the same sign. Indeed, from (12) it follows that $|\overline{r}/p(x) - 1| \leq \mathbf{u} + \gamma_{2n}^2\,\mathrm{cond}(p,x)$, and therefore $\overline{r}/p(x) \geq 1 - \mathbf{u} - \gamma_{2n}^2\,\mathrm{cond}(p,x)$. But (15) implies that $1 - \mathbf{u} - \gamma_{2n}^2\,\mathrm{cond}(p,x) > 1 - 3\mathbf{u}/(2+\mathbf{u}) > 0$, hence $\overline{r}/p(x) > 0$. Since $\overline{r}$ and $p(x)$ have the same sign, it is easy to see that

$$(1-\mathbf{u})|p(x)| - \gamma_{2n}^2\,\widetilde{p}(x) \leq |\overline{r}|. \tag{16}$$

Indeed, if $p(x) > 0$ then (12) implies $p(x) - \mathbf{u}|p(x)| - \gamma_{2n}^2\,\widetilde{p}(x) \leq \overline{r} = |\overline{r}|$. If $p(x) < 0$, from (12) it follows that $\overline{r} \leq p(x) + \mathbf{u}|p(x)| + \gamma_{2n}^2\,\widetilde{p}(x)$, hence $-p(x) - \mathbf{u}|p(x)| - \gamma_{2n}^2\,\widetilde{p}(x) \leq -\overline{r} = |\overline{r}|$.

Next, a small computation proves that

$$\mathrm{cond}(p,x) < \frac{1-\mathbf{u}}{2+\mathbf{u}}\mathbf{u}{\gamma_{2n}}^{-2}$$

if and only if

$$\gamma_{2n}^2\,\widetilde{p}(x) < \frac{\mathbf{u}}{2}\left[(1-\mathbf{u})|p(x)| - \gamma_{2n}^2\,\widetilde{p}(x)\right].$$

Finally, from (13) and (16) it follows

$$|\widehat{c} - c| \leq \gamma_{2n}^2\,\widetilde{p}(x) < \frac{\mathbf{u}}{2}\left[(1-\mathbf{u})|p(x)| - \gamma_{2n}^2\,\widetilde{p}(x)\right] \leq \frac{\mathbf{u}}{2}|\overline{r}|.$$

From Lemma 6 we deduce that $\overline{r}$ is faithfully rounded. □

Numerical values for the upper bound (15) to ensure faithful rounding with the compensated Horner algorithm are presented in Table 1 for degrees varying from 10 to 500. We assume that the working precision is the IEEE-754 double precision. For example, when evaluating a polynomial of degree 100, we know from Table 1 that CompHorner (p,x) is a faithful rounding of $p(x)$ as long as $\mathrm{cond}(p,x) < 1.13 \cdot 10^{11}$.

Table 1. *A priori* **bounds on the condition number w.r.t. polynomial degree n.**

n	10	100	200
$\frac{1-\mathbf{u}}{2-\mathbf{u}}\mathbf{u}\gamma_{2n}{}^{-2}$	$1.13 \cdot 10^{13}$	$1.13 \cdot 10^{11}$	$2.82 \cdot 10^{10}$

4 Dynamic and validated error bounds for faithful rounding and accuracy

The results presented in Section 3 are perfectly suited for theoretical purpose, for instance when we can *a priori* bound the condition number of the evaluation. However, neither the error bound in Theorem 4, nor the criterion proposed in Theorem 7 can be easily checked using only floating point arithmetic. Here we provide dynamic counterparts of Theorem 4 and Proposition 7, that can be evaluated using floating point arithmetic in the "round to the nearest" rounding mode.

Lemma 8. *Consider a polynomial p of degree n with floating point coefficients, and x a floating point value. We use the notations of Algorithm 3, and we denote $(p_\pi + p_\sigma)(x)$ by c. Then*

$$|c - \widehat{c}| \leq \mathrm{fl}\left(\frac{\widehat{\gamma}_{2n-1}\mathsf{Horner}\left(|p_\pi \oplus p_\sigma|, |x|\right)}{1 - 2(n+1)\mathbf{u}}\right) := \widehat{\alpha}. \quad (17)$$

Proof. Let us denote $\mathsf{Horner}\left(|p_\pi \oplus p_\sigma|, |x|\right)$ by $\widehat{b}$. Since $c = (p_\pi + p_\sigma)(x)$ and $\widehat{c} = \mathsf{Horner}\left(p_\pi \oplus p_\sigma, x\right)$ where p_π and p_σ are two polynomials of degree $n-1$, Lemma 1 yields

$$|c - \widehat{c}| \leq \gamma_{2n-1}(\widetilde{p_\pi + p_\sigma})(x) \leq (1+\mathbf{u})^{2n-1}\gamma_{2n-1}\,\widehat{b}.$$

From (4) and (3) it follows that

$$|c - \widehat{c}| \leq (1+\mathbf{u})^{2n}\,\widehat{\gamma}_{2n-1}\,\widehat{b} \leq (1+\mathbf{u})^{2n+1}\,\mathrm{fl}(\widehat{\gamma}_{2n-1}\,\widehat{b}).$$

Finally we use (5) to obtain the error bound (17). $\square$

Remark 2. Lemma 8 allows us to compute a validated error bound for the computed correcting term $\widehat{c}$. We apply this result twice to derive next Theorem 9. First with Lemma 6 it yields the expected dynamic condition for faithful rounding. Then from the EFT for the Horner algorithm (Theorem 3) we know that $p(x) = \widehat{r} + c$. Since $\overline{r} = \widehat{r} \oplus \widehat{c}$, we deduce $|\,\overline{r} - p(x)| = |(\,\widehat{r} \oplus \widehat{c}) - (\,\widehat{r} + \widehat{c}) + (\,\widehat{c} - c)|$. Hence we have

$$|\,\overline{r} - p(x)| \leq |(\,\widehat{r} \oplus \widehat{c}) - (\,\widehat{r} + \widehat{c})| + |(\widehat{c} - c)|. \quad (18)$$

The first term $|(\,\widehat{r} \oplus \widehat{c}) - (\,\widehat{r} + \widehat{c})|$ in the previous inequality is basically the absolute rounding error that occurs when computing $\overline{r} = \widehat{r} \oplus \widehat{c}$. Using only the bound (3) of the standard model of floating point arithmetic, it could be bounded by $\mathbf{u}|\,\overline{r}|$. But here we benefit again from error free transformations using algorithm TwoSum to compute exactly the actual rounding error, which leads to a sharper error bound. Next Relation (19) improves the dynamic bound presented in [3].

Theorem 9. *Consider a polynomial p of degree n with floating point coefficients, and x a floating point value. Let $\overline{r}$ be the computed value, $\overline{r} = \mathsf{CompHorner}\,(p,x)$ (Algorithm 3) and let $\widehat{\alpha}$ be the error bound defined by Relation (17).*

- *If $\widehat{\alpha} < \frac{\mathbf{u}}{2}|\,\overline{r}|$, then $\overline{r}$ is a faithful rounding of $p(x)$.*
- *Let e be the floating point value such that $\overline{r} + e = \widehat{r} + \widehat{c}$, i.e., $[\,\overline{r}, e] = \mathsf{TwoSum}\,(\,\widehat{r}, \widehat{c})$, where $\widehat{r}$ and $\widehat{c}$ are defined by Algorithm 3. The absolute error of the computed result $\overline{r} = \mathsf{CompHorner}\,(p,x)$ is bounded as follows,*

$$|\,\overline{r} - p(x)| \leq \mathrm{fl}\left(\frac{\widehat{\alpha} + |e|}{1 - 2u}\right) := \widehat{\beta}. \quad (19)$$

Proof. The first proposition follows directly from Lemma 6.

By hypothesis $\overline{r} = \widehat{r} + \widehat{c} - e$, and from Theorem 3 we have $p(x) = \widehat{r} + c$, thus

$$|\,\overline{r} - p(x)| = |\,\widehat{c} - c - e| \leq |\,\widehat{c} - c| + |e| \leq \widehat{\alpha} + |e|.$$

From (3) and (5) it follows that

$$|\,\overline{r} - p(x)| \leq (1+\mathbf{u})\,\mathrm{fl}(\,\widehat{\alpha} + |e|) \leq \mathrm{fl}\left(\frac{\widehat{\alpha} + |e|}{1 - 2u}\right);$$

which proves the second proposition. $\square$

From Theorem 9 we deduce the expected algorithm. It computes the compensated result $\overline{r}$ together with the validated error bound $\widehat{\beta}$. Moreover, the boolean value $\mathsf{isfaithful}$ is set to true if and only if the result is proved to be faithfully rounded — if $\mathsf{isfaithful}$ is set to false, then $\overline{r}$ may or may not be a faithful rounding of $p(x)$.

When the check for faithful rounding fails (the boolean $\mathsf{isfaithful}$ is false), $\overline{r}$ may or may not be a faithful rounding of $p(x)$, but the error in the $\overline{r}$ is still bounded by $\widehat{\beta}$. Nevertheless, the computation of the error bound $\widehat{\beta}$ can be safely omitted in the previous algorithm $\mathsf{CompHornerIsFaithful}$ since $\mathsf{isfaithful}$ does not depend on $\widehat{\beta}$.

5 Experimental results

We consider polynomials with floating point coefficients and floating point entries x. We use Matlab codes for $\mathsf{CompHorner}$ (Algorithm 3) and $\mathsf{CompHornerIsFaithful}$ (Algorithm 4) within the accuracy tests we propose hereafter.

Algorithm 4 Compensated Horner algorithm with check of the faithful rounding

```
function $[\overline{r},\ \widehat{\beta}, \mathsf{isfaithful}] = \mathsf{CompHornerIsFaithful}\,(p, x)$
    $[\widehat{r}, p_\pi, p_\sigma] = \mathsf{EFTHorner}\,(p, x)$
    $\widehat{c} = \mathsf{Horner}\,(p_\pi \oplus p_\sigma, x)$
    $\widehat{b} = \mathsf{Horner}\,(|p_\pi \oplus p_\sigma|, |x|)$
    $\widehat{\alpha} = (\widehat{\gamma}_{2n-1} \otimes \widehat{b}) \oslash (1 \ominus 2(n+1) \otimes \mathbf{u})$
    $[\overline{r}, e] = \mathsf{TwoSum}\,(\widehat{r},\ \widehat{c})$
    $\widehat{\beta} = (\widehat{\alpha} \oplus |e|) \oslash (1 - 2 \otimes \mathbf{u})$
    $\mathsf{isfaithful} = (\widehat{\alpha} < \frac{\mathbf{u}}{2}|\overline{r}|)$
```

CompHorner requires $21n + \mathcal{O}(1)$ flop and that CompHornerIsFaithful requires $26n + \mathcal{O}(1)$ flop. For testing the time performances, the previous algorithms are coded in C language and several test platforms are described in next Table 2.

5.1 Accuracy tests

We focus on both the *a priori* and dynamic bounds with two sets of tests. We recall that two cases may occur when the dynamic test for faithful rounding in Algorithm 4 is performed.

1. If the dynamic test is satisfied, this proves that the compensated result is a faithful rounding of the exact $p(x)$. Corresponding plots are reported with a square (□) in Figure 1 and Figure 2.

2. If the dynamic test fails then the compensated result may or may not be faithfully rounded. We distinguish two sub-cases where we compare the compensated results to reference ones obtained from high-precision computation.

(a) If the compensated result is actually faithfully rounded, the evaluation value is a filled circle (•).

(b) Otherwise the compensated result is not a faithful rounding of the exact $p(x)$ and we plot a cross (×).

We consider huge condition numbers in the following tests. This have a sense here since the entries and the coefficients of every tested polynomial are floating point numbers.

5.1.1 Faithful rounding with compensated Horner

In the first experiment, we evaluate the expanded form of polynomials $p_n(x) = (1-x)^n$, for degree $n = 6$ and 8, in the neighborhood of the multiple root $x = 1$. These evaluations are extremely ill-conditioned since $\mathrm{cond}(p_n, x) = |(1+|x|)/(1-x)|^n$. For a given degree, this condition number is arbitrarily large as the entry x tends closer to the root 1. These condition numbers are plotted in the lower frames of Figure 1 while x varies around the root. The well known relation between the lost of accuracy and the nearness and the multiplicity of the root, *i.e.*, the increasing of the condition number, is clearly illustrated. Evaluation is no more faithfully rounded for entries too close to the root (evaluations for entries out of the range of the x-axis on Figure 1 are faithfully rounded). Alas the dynamic bound fails to identify every faithful result and so is pessimistic, even more and more pessimistic as the condition number increases.

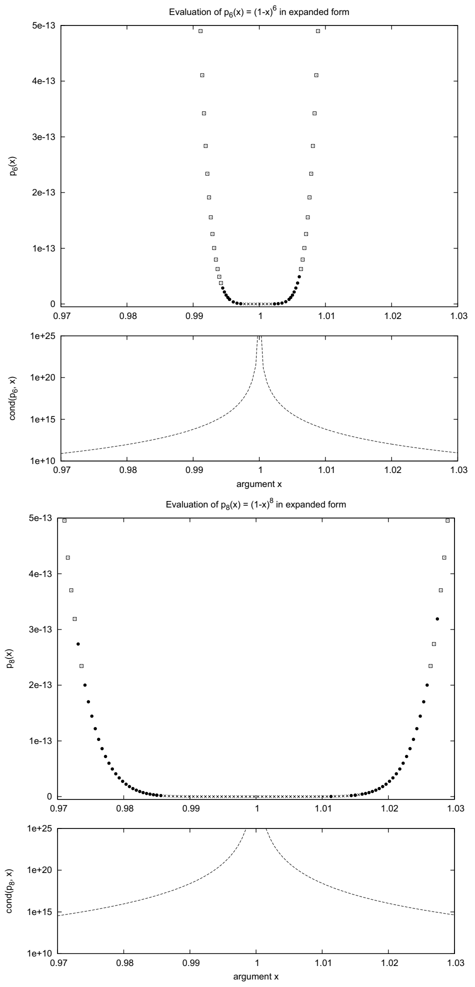

Figure 1. Accuracy tests for $(1-x)^n$ near $x = 1$ with CompHornerIsFaithful. See Subsection 5.1 for the definition of displayed plots.

For the next experiment, we first designed a generator of arbitrarily ill-conditioned polynomial evaluations. It relies on definition (6) of the condition number. Given a degree n, a floating point entry x and a targeted value C for the condition number, it generates a polynomial p with floating

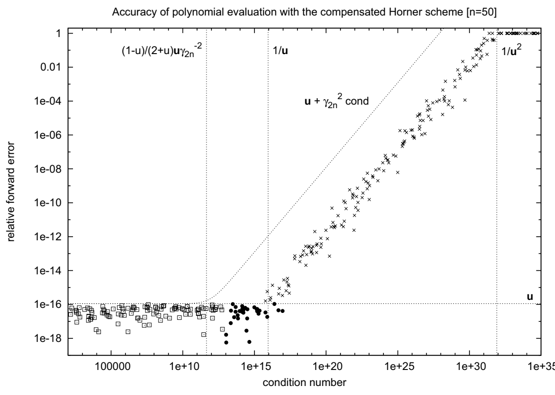

Figure 2. Accuracy of CompHornerIsFaithful w.r.t. to the condition number. Leftmost vertical line is the *a priori* sufficient condition (15) and broken line is the *a priori* bound (14).

point coefficients such that $\mathrm{cond}(p, x)$ has the same order of magnitude as C. The principle of the generator is the following: $\lfloor n/2 \rfloor$ floating point coefficients of p are randomly generated such that $\widetilde{p}(x) = \sum |a_i||x|^i \approx C$, and then the remaining coefficients are generated ensuring $|p(x)| \approx 1$ thanks to high accuracy computation. Therefore we obtain polynomials p such that $\mathrm{cond}(p, x) = \widetilde{p}(x)/|p(x)| \approx C$, for arbitrary values of C.

In this test we generate polynomials of degree 50 whose condition numbers vary from about 10^2 to 10^{35}. The results of the tests performed with CompHornerIsFaithful (Algorithm 4) are reported on Figure 2. On this figure the horizontal axis does not represent anymore the x entry range but the condition number Relation (6).

We observe that the compensated algorithm exhibits the expected behavior. The relative error in the compensated result is smaller than the working precision $\mathbf{u}$ —the horizontal line— as long as the condition number is smaller than $1/\mathbf{u}$ —the second vertical line. Then, for condition numbers between $1/\mathbf{u}$ and $1/\mathbf{u}^2$, this relative error degrades to no accuracy at all. As usual, the *a priori* error bound (14) appears to be pessimistic by many orders of magnitude —compare the observed behavior with the comments we provide just after Relation (14)

The *a priori* sufficient condition (15) for faithful rounding with respect to the condition number is also represented on Figure 2 —the leftmost vertical line. As expected, every polynomial evaluation with a condition number smaller than this *a priori* bound (15) is faithfully evaluated with Algorithm 4. We also see that the dynamic test for faithful rounding (Proposition 9) succeeds for condition numbers larger than the *a priori* bound (15) —let us recall that all the compensated evaluations proved to be faithfully rounded thanks to the dynamic test are reported with a square. Finally we notice that the compensated Horner algorithm produces accurate evaluations for condition numbers up to about $1/\mathbf{u}$ —evaluations reported with a square or a filled circle.

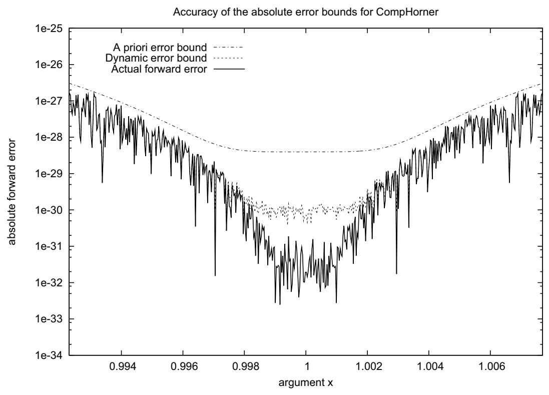

Figure 3. Significance of the error bounds.

5.1.2 Significance of the dynamic error bound

We illustrate the significance of the dynamic error bound (19), compared to the *a priori* absolute error bound (12) and to the actual forward error. We evaluate the expanded form of $p(x) = (1 - x)^5$ for 400 points near $x = 1$. For each value of the entry x, we compute CompHorner (p, x) (Algorithm 3), the associated dynamic error bound (19) and the actual forward error. The results are reported on Figure 3.

As already noticed, the closer the argument is to the root 1 (*i.e.*, the more the condition number increases), the more pessimistic becomes the *a priori* error bound. Our dynamic error bound is more significant than the *a priori* error bound as it takes into account the rounding errors that occur during the computation.

5.2 Time performances

All experiments are performed using IEEE-754 double precision. Since the double-doubles [9] are usually considered as the most efficient portable library to double the IEEE-754 double precision, we consider it as a reference in the following comparisons. For our purpose, it suffices to know that a double-double number a is the pair (a_h, a_l) of IEEE-754 floating point numbers with $a = a_h + a_l$ and $|a_l| \leq \mathbf{u}|a_h|$. This property implies a renormalisation step after every arithmetic operation with double-double values. We denote by DDHorner our implementation of the Horner algorithm with the double-double format, derived from the implementation proposed in [9].

We implement the three algorithms CompHorner, CompHornerIsFaithful and DDHorner in a C code to measure their overhead compared to the Horner algorithm.

Table 2. Measured time performances.

		CompHorner / Horner	CHIsFaithful / Horner	DDHorner / Horner
P4	gcc 3.3.5	3.77	5.52	10.00
	icc 9.1	3.06	5.31	8.88
AMD64	gcc 4.0.1	3.89	4.43	10.48
IA'64	icc 3.4.6	3.64	4.59	5.50
	icc 9.1	1.87	2.30	8.78
		$\sim 2-4$	$\sim 4-6$	$\sim 5-10$

We program these tests straightforwardly with no other optimization than the ones performed by the compiler. All timings are done with the cache warmed to minimize the memory traffic over-cost.

We test the running times of these algorithms for different architectures with different compilers as described in Table 2. Our measures are performed with polynomials whose degree vary from 5 to 200 by step of 5. For each algorithm, we measure the ratio of its computing time over the computing time of the classic Horner algorithm; we display the average time ratio over all test cases in Table 2.

The results presented in Table 2 show that the slowdown factor introduced by CompHorner compared to the classic Horner roughly varies between 2 and 4. The same slowdown factor varies between 4 and 6 for CompHornerIsFaithful and between 5 and 10 for DDHorner. We can see that CompHornerIsFaithful runs at most 2 times slower than CompHorner: the over-cost due to the dynamic test for faithful rounding is therefore quite reasonable. Anyway CompHorner and CompHornerIsFaithful run both significantly faster than DDHorner.

We provide time ratios for IA'64 architecture (Itanium 2). Tested algorithms take benefit from IA'64 instructions, *e.g.*, fma, but are not described here —see [7] for details.

6 Conclusion

Compensated Horner algorithm yields more accurate polynomial evaluation than the classic Horner iteration. Its accuracy is similar to a Horner iteration performed in a doubled working precision. Hence compensated Horner may perform a faithful polynomial evaluation with IEEE-754 floating point arithmetic in the "round to the nearest" rounding mode. An *a priori* sufficient condition with respect to the condition number that ensures such faithfulness has been defined thanks to the error free transformations.

These error free transformations also allow us to derive a dynamic sufficient condition that is more significant to check for faithful rounding with CompHorner.

It is interesting to remark here that the significance of this dynamic bound can be improved easily. Whereas bounding the error in the computation of the (polynomial) correcting term in Relation (17), a good approximate of the actual error could be computed (applying again CompHorner to the correcting term). Of course such extra computation will introduce more running time while such overhead is not always useful. So it suffices to run this extra (but costly) checking only if the previous dynamic one fails —a similar strategy as in dynamic filters for geometric algorithms.

Compared to the classic Horner algorithm, experimental results exhibit reasonable over-costs for accurate polynomial evaluation (between 2 and 4) and even for this computation with a dynamic checking for faithfulness (between 4 and 6). Let us finally remark than such computation that provides as accuracy as if the working precision is doubled and a faithfulness checking costs no more running time than the "double-double" counterpart without any check.

Future work will be to consider subnormal results and also an adaptive algorithm that ensure faithful rounding for polynomials with an arbitrary condition number.

References

[1] T. J. Dekker. A floating-point technique for extending the available precision. *Numer. Math.*, 18:224–242, 1971.

[2] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[3] S. Graillat, P. Langlois, and N. Louvet. Compensated Horner scheme. Technical report, Univ. of Perpignan, France, 2005.

[4] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, second edition, 2002.

[5] *IEEE Standard for binary floating-point arithmetic, ANSI/IEEE Standard 754-1985*. 1985.

[6] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*. Addison-Wesley, third edition, 1998.

[7] P. Langlois and N. Louvet. Operator dependant compensated algorithms. In *Proceedings of the 12th GAMM - IMACS - SCAN, Duisburg, Germany*, 2007.

[8] C. Li, S. Pion, and C.-K. Yap. Recent progress in exact geometric computation. *Journal of Logic and Algebraic Programming*, 64(1):85–111, 2005.

[9] X. S. Li, J. W. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Y. Kang, A. Kapur, M. C. Martin, B. J. Thompson, T. Tung, and D. J. Yoo. Design, implementation and testing of extended and mixed precision BLAS. *ACM Trans. Math. Software*, 28(2):152–205, 2002.

[10] P. Markstein. *IA-64 and elementary functions: speed and precision*. Prentice-Hall, 2000.

[11] J.-M. Muller. *Elementary functions: algorithms and implementation*. Birkhäuser, second edition, 2006.

[12] T. Ogita, S. M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM J. Sci. Comput.*, 26(6):1955–1988, 2005.

[13] D. M. Priest. Algorithms for arbitrary precision floating point arithmetic. In *Proceedings of IEEE ARITH-10*, pages 132–144, 1991.

[14] S. M. Rump, T. Ogita, and S. Oishi. Accurate summation. Technical report, T.U. Hamburg, Germany, 2005.